



Digitize the Official Lebanese Journal

Final report

Bachour Peter
Creidi Kevin

18/01/2019

Professor:
Abou Jaoude Sassine

Table of Content

| | | |
|-------|-------------------------------------|----|
| 1 | Abstract | 4 |
| 2 | Introduction..... | 5 |
| 2.1 | Context | 5 |
| 2.2 | Subject..... | 5 |
| 2.3 | Objectives | 5 |
| 2.3.1 | Optical character recognition..... | 5 |
| 2.3.2 | Database..... | 5 |
| 2.3.3 | Search | 5 |
| 3 | Analysis / State of the art | 6 |
| 3.1 | OCR.Space | 6 |
| 3.2 | An-Nahar | 6 |
| 3.3 | Summary..... | 6 |
| 4 | Technical environment..... | 7 |
| 4.1 | IDE and Database Server | 7 |
| 4.2 | Front-End..... | 7 |
| 4.3 | MVC Framework..... | 7 |
| 4.4 | 3 Tier Architecture..... | 8 |
| 4.4.1 | Presentation Layer..... | 8 |
| 4.4.2 | Business Logic Layer | 8 |
| 4.4.3 | Data Access Layer | 8 |
| 4.5 | Source Manager | 8 |
| 4.6 | Project Management..... | 8 |
| 5 | Architecture..... | 9 |
| 5.1 | Model View Controller pattern | 9 |
| 5.2 | Database..... | 9 |
| 5.3 | Mockup of the WebApp | 9 |
| 6 | Development | 11 |
| 6.1 | Front-End..... | 11 |
| 6.1.1 | Bootstrap..... | 11 |
| 6.1.2 | Web Application Design | 11 |
| 6.2 | Back-End | 13 |
| 6.2.1 | Entities..... | 13 |
| 6.2.2 | Repositories..... | 13 |
| 6.2.3 | Services..... | 14 |
| 6.2.4 | Controller..... | 14 |
| 6.2.5 | Role-Based Authorization..... | 15 |
| 6.3 | Database..... | 16 |
| 6.4 | Swagger | 17 |
| 6.1 | Facebook and Twitter API | 17 |
| 7 | Discussion and Future Work..... | 18 |
| 7.1 | Discussion | 18 |
| 7.1.1 | Difficulty 1 | 18 |
| 7.1.2 | Difficulty 2 | 18 |

| | | |
|-------|----------------------|----|
| 7.1.3 | Difficulty 3 | 18 |
| 7.2 | Future Work | 18 |
| 7.2.1 | User Interface | 18 |
| 7.2.2 | Favorite Pages | 18 |
| 8 | Conclusion | 18 |
| 9 | Glossary | 19 |

| | |
|--|----|
| Figure 1 - Screenshot of OCR.Space | 6 |
| Figure 2 - An-Nahar search engine | 6 |
| Figure 3 - Data in Database | 9 |
| Figure 4 - Home Page Mockup | 9 |
| Figure 5 - Search Page Mockup | 10 |
| Figure 6 - Web Application Home Page..... | 11 |
| Figure 7 - Web Application search page..... | 11 |
| Figure 8 - Register Page | 12 |
| Figure 9 - Login Page | 12 |
| Figure 10 - Upload Page for Admins only | 12 |
| Figure 11 - Class for page entity | 13 |
| Figure 12 - Repository referring to the Page | 13 |
| Figure 13 - Service referring to the entity Page | 14 |
| Figure 14 - Functions in the Page Controller | 14 |
| Figure 15 - Authorizing logged in Users only..... | 15 |
| Figure 16 - Restricting this Page only to this User..... | 15 |
| Figure 17 - User Logged in | 15 |
| Figure 18 - Migration to SQL Server | 16 |
| Figure 19 - Filling the table | 16 |
| Figure 20 - Swagger in ConfigureServices | 17 |
| Figure 21 - Facebook and Twitter APIs..... | 17 |
| Figure 22 - Facebook and Twitter Buttons | 17 |

1 Abstract

This report describes a project called “Digitize the Official Lebanese Journal” Saint Joseph University of Beirut as part of our final year project. This Project is about designing a web interface, the purpose of which is to allow the user to search the official Lebanese Journal. This interface is user friendly since the user can upload and read data easily and fast.

2 Introduction

2.1 Context

The official newspaper of Lebanon out weekly contains the laws and official announcements of the Government of Lebanon. It gives all the necessary news pictures and reports to the readers to benefit and be up to date on everything happening in Lebanon.

A part already exists in digital soft [copy](#), which the government allows to subscribe and search, but another part remains to be scanned. Why? Because all the readable news is given in pictures, therefore the reader cannot edit or search in the text.

2.2 Subject

The goal is to create a web application, which allows to download and scan images from this log, and to create an opportunity to subscribe and search and find in the scanned material. Plus, it will make sure to remember the user's favorite pages for a later read. We are creating a real full text search that allows the users to scan full texts and point to the right scanned file without being bothered. We are making the way of reading a journal more digital and surely faster, easier and more comfortable. Therefore, we give guarantee of the best experience to read newspapers nowadays.



2.3 Objectives

2.3.1 Optical character recognition

At first, implementing an API than convert easily the picture into a text through Optical character recognition.

2.3.2 Database

Using Code First in C# Visual Studio 2017, inserting into the database all the required data for each page.

2.3.3 Search

Create a full text search that can return the page requested by the user.

3 Analysis / State of the art

There has been worldwide growth in optical character recognition and many newspapers are using it in order to create to their customers a searchable engine.

3.1 OCR.Space¹

This online optical character recognition software supports many languages. Without installation on your desktop, this software converts your pdf into many output formats like Microsoft Word (docx), Microsoft Excel (xlsx) and text plain (txt). The output format look exactly like the original and can be editable.

Free Online OCR - Convert images and PDF to text (Powered by the [OCR API](#))

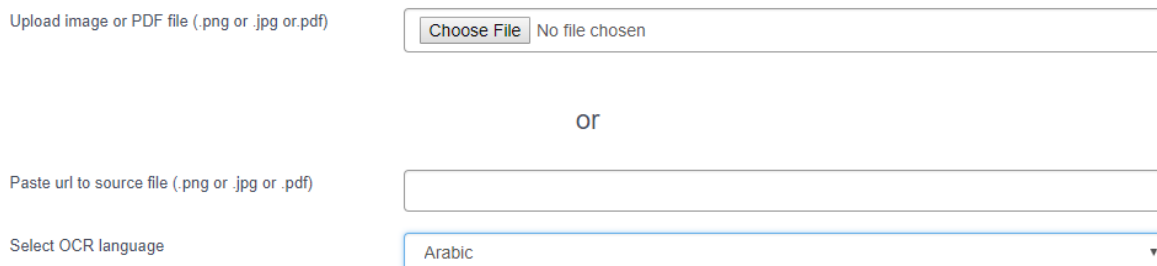


Figure 1 - Screenshot of OCR.Space

3.2 An-Nahar

An-Nahar² is a Leading Arabic language daily newspaper published in Lebanon. An-Nahar has its own webpage where you can search for any news you want. This news can be recent or old, the search engine will still find the user's request.

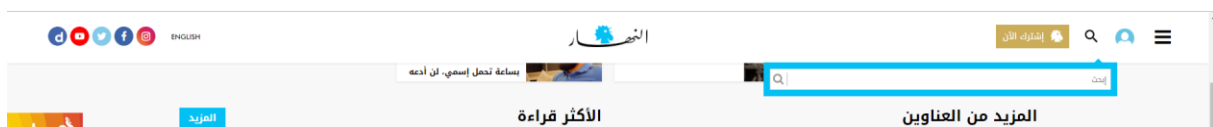


Figure 2 - An-Nahar search engine

3.3 Summary

Multiple available software allows the user to convert his pdf or images into text. Moreover, many newspapers have a search engine built-in in their web apps. However, none of them directly address the subject of this project. Therefore, this project will be a novelty since it will create a web interface that can have these two technologies built-in.

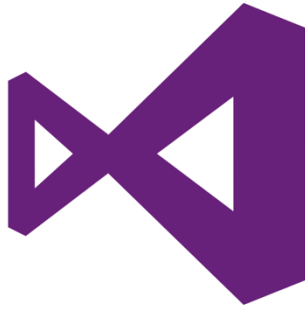
¹ <https://ocr.space/>

² <https://www.annahar.com/>

4 Technical environment

4.1 IDE and Database Server

Visual Studio 2017 is an IDE to develop C# applications. It also has the .Net Core built in. It is the best fit to develop .Net applications. As for the database server, SQL Server 2017 is being used to store objects since it is an SQL database that works well with visual studio.

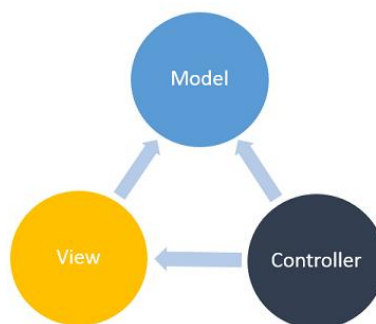


4.2 Front-End

With the logic of the application being written with the help C#, we need an interface where the user can communicate with the web application. The interface will be put in place using HTML, CSS, CSHTML, JavaScript and Bootstrap. Bootstrap allows creating responsive web pages, all the while being a free and open source library that can help design the pages.

4.3 MVC Framework

Multiple available frameworks can easily help build such a system. They go from Node.js, that shines in building fast, scalable network applications to Django that relies on Python and is secure, scalable versatile and fast. We decided to use the MVC (Model View Controller) framework for multiple reasons. MVC is lightweight with respect to size and transparency, it also can provide multiple views. Moreover, code duplication is very limited because MVC separates data and business logic from the display.



4.4 3 Tier Architecture

A 3-tier architecture is a type of software architecture which is composed of three “tiers” or “layers” of logical computing. These 3 tiers are: Presentation Layer, Business Logic Layer, Data Access Layer.

4.4.1 Presentation Layer

The presentation tier is the front-end layer in the 3-tier system and consists of the user interface. This user interface is often a graphical one accessible through a web browser or web-based application and which displays content and information useful to an end user.

4.4.2 Business Logic Layer

The Business Logic tier contains the functional business logic which drives an application’s core capabilities.

4.4.3 Data Access Layer

The data access layer provides simplified access to data stored in persistent storage.

4.5 Source Manager

We created a Git repository in GitHub in order to have a version control where we can collaborate with our supervisor. Since Git uses command line interface, we decided to download Tortoise Git³ to make it easier to push and pull.



4.6 Project Management

In order to track our work, we used Trello.



³ <https://tortoisegit.org>

5 Architecture

5.1 Model View Controller pattern

.Net Core relies on the MVC design pattern. This pattern is used to separate the application into 3 layers with different responsibilities. The model represents the logic of the application and is written in C#. The view represents the visualization of the data that the model contains, and is consisted of HTML, CSS, and JavaScript pages. The controller acts on both model and view. It controls the data flow into model objects and updates the view whenever data changes. It keeps view and model separate.

5.2 Database

In order to make it easier for the search, we thought about inserting both, the text and the image with their issue dates and issue numbers and not to forget the page number.

| Results | | Messages | | | | |
|---------|----|------------|-------------------------------------|-----------------------------|-------------|--|
| | Id | PageNumber | FullText | IssuDate | IssueNumber | Image |
| 1 | 20 | 4467 | ***** Result for Image/Page 1 ***** | 2018-09-27 00:00:00.0000000 | 42 | 0x47494638396104025803F70000FFFFFF000000F7F7FEF... |
| 2 | 21 | 4468 | ***** Result for Image/Page 1 ***** | 2018-09-27 00:00:00.0000000 | 42 | 0x47494638396104025803F70000FFFFFF000000F7F7FEF... |
| 3 | 22 | 4469 | ***** Result for Image/Page 1 ***** | 2018-09-27 00:00:00.0000000 | 42 | 0x47494638396104025803F70000FFFFFF000000F7F7FEF... |
| 4 | 23 | 4470 | ***** Result for Image/Page 1 ***** | 2018-09-27 00:00:00.0000000 | 42 | 0x47494638396104025803F70000FFFFFF000000F7F7FEF... |
| 5 | 24 | 4471 | ***** Result for Image/Page 1 ***** | 2018-09-27 00:00:00.0000000 | 42 | 0x47494638396104025803F70000FFFFFF000000F7F7FEF... |
| 6 | 25 | 4472 | ***** Result for Image/Page 1 ***** | 2018-09-27 00:00:00.0000000 | 42 | 0x47494638396104025803F70000FFFFFF000000F7F7FEF... |
| 7 | 26 | 4473 | ***** Result for Image/Page 1 ***** | 2018-09-27 00:00:00.0000000 | 42 | 0x47494638396104025803F70000FFFFFF000000F7F7FEF... |
| 8 | 27 | 4474 | ***** Result for Image/Page 1 ***** | 2018-09-27 00:00:00.0000000 | 42 | 0x47494638396104025803F70000FFFFFF000000F7F7FEF... |
| 9 | 28 | 4475 | ***** Result for Image/Page 1 ***** | 2018-09-27 00:00:00.0000000 | 42 | 0x47494638396104025803F70000FFFFFF000000F7F7FEF... |

Figure 3 - Data in Database

5.3 Mockup of the WebApp

We prepared the design of the WebApp before scripting it by using ninja mockup⁴.

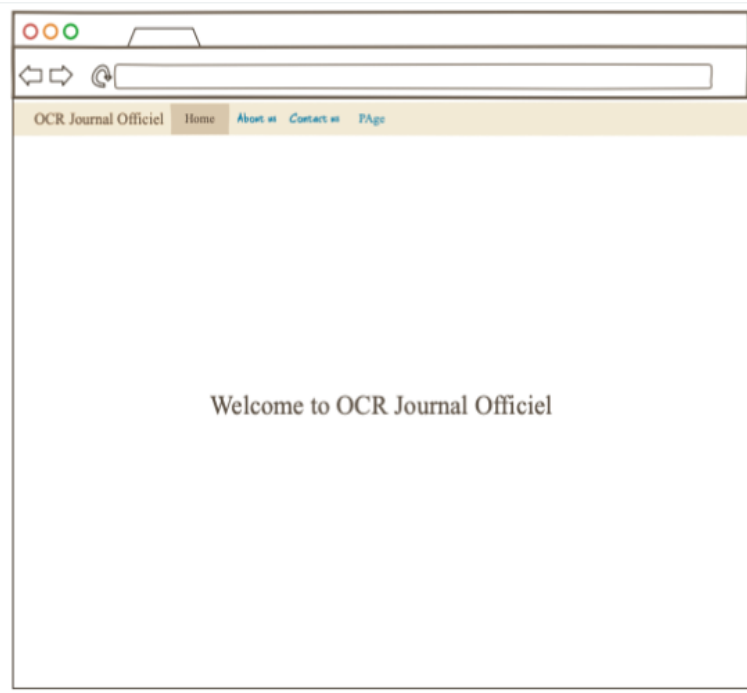


Figure 4 - Home Page Mockup

⁴ <https://ninjamock.com/s/VNXQMTx>



Figure 5 - Search Page Mockup

6 Development

6.1 Front-End

6.1.1 Bootstrap

The design of the web application was done using bootstrap which contains HTML, CSS and JavaScript libraries.

6.1.2 Web Application Design



Figure 6 - Web Application Home Page

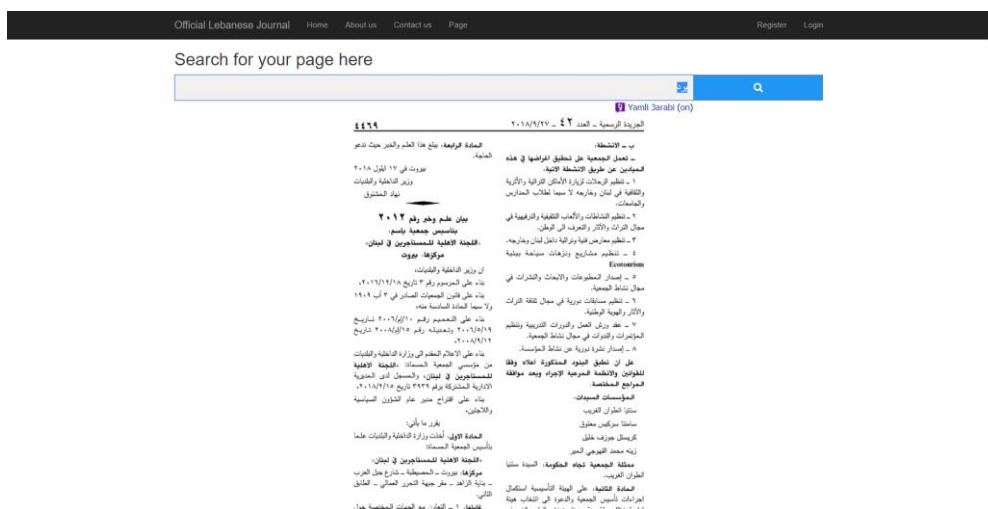


Figure 7 - Web Application search page

Official Lebanese Journal Home Page About us Contact us Register Login

Register

Create a new account.

Email

Full screen Sign

Password

Confirm password

Register

© 2019 - Official Lebanese Journal

Figure 8 - Register Page

Official Lebanese Journal Home Page About us Contact us Register Login

Log in

Use a local account to log in.

Email

Full screen Sign

Password

☐ Remember me?

Log in

Use another service to log in.

There are no external authentication services configured. See [this article](#) for details on setting up this ASP.NET application to support logging in via external services.

[Forgot your password?](#)

[Register as a new user](#)

© 2019 - Official Lebanese Journal

Figure 9 - Login Page

Official Lebanese Journal Home Page About us Contact us Upload Hello peter_bachour@hotmail.fr Logout

Upload your new page here

Page number...

Tweet

Share

© 2019 - Official Lebanese Journal

Figure 10 - Upload Page for Admins only

6.2 Back-End

6.2.1 Entities

An entity is a persistent object stocked in the database. All the data of this object can be easily accessible through the .Net Framework MVC.

```
public class Page
{
    [Key]
    public int Id { get; set; }

    public int PageNumber { get; set; }

    public string FullText { get; set; }

    public DateTime IssuDate { get; set; }

    public int IssueNumber { get; set; }

    [Column(TypeName = "varbinary(max)")]
    public byte[] Image { get; set; }
}
```

Figure 11 - Class for page entity

6.2.2 Repositories

The web application should insert, modify or delete objects from the database. Therefore, adding functions is necessary.

```
public class PagesRepository : Repository<Page> , IPagesRepository
{
    public PagesRepository(ApplicationDbContext _db) : base(_db)
    {
    }

    public async Task<List<Tuple<int, string>>> GetDistinctByTextAsync(string searchText, CancellationToken ct) ...

    public async Task<Page> GetAsync(int id, CancellationToken ct) ...
}
```

Figure 12 - Repository referring to the Page

6.2.3 Services

The repository that we have mentioned earlier offers many functions for the developer that not all of them will be used. That is why we have created services, one for each entity. Every service has an instance of the correspondent repository.

```
public class PageService : BaseService, IPageService
{
    public PageService(IUnitOfWork uow, ILogger<PageService> logger, IModelMapHelper mapper) ...
    private async Task<bool> UploadPage(string txtName, string imgName, int pageNumber, CancellationToken ct) ...
    public async Task<bool> UploadPages(List<PageUploadDto> pages, CancellationToken ct) ...
    public async Task<bool> UploadNewPage(int pageNumber, CancellationToken ct) ...
    public async Task<byte[]> GetDummyImage(int imageId, CancellationToken ct) ...
    public async Task<List<Tuple<int, string>>> SearchForText(string text, CancellationToken ct) ...
}
```

Figure 13 - Service referring to the entity Page

6.2.4 Controller

The controller creates a link between the front-end and the back-end. In fact, every request of page done by a HTML page will be delegated by a function in the controller.

```
[Route("[controller]/[action]")]
public class PageController : BaseController<PageController>
{
    private readonly IPageService _pageService;
    public PageController(
        IPageService pageService,
        IBaseservice baseService,
        ILogger<PageController> logger) ...

    [HttpPost]
    [AllowAnonymous]
    public async Task<ActionResult> Upload(CancellationToken ct) ...

    [HttpPost]
    [AllowAnonymous]
    public async Task<ActionResult> UploadNew(int pageNumber, CancellationToken ct) ...

    [AllowAnonymous]
    [HttpGet]
    public IActionResult Index() ...

    [HttpGet]
    public IActionResult Upload() ...

    [AllowAnonymous]
    [HttpGet]
    public async Task<ActionResult> GetDummyImage(int id, CancellationToken ct) ...

    [HttpPost]
    [AllowAnonymous]
    public async Task<ActionResult> SearchForText(string search, CancellationToken ct) ...
}
```

Figure 14 - Functions in the Page Controller

6.2.5 Role-Based Authorization

In order to restrict the system access to authorized users only which means that users that are signed in only we use the [Authorize()] tag :

```
[Authorize()]
public IActionResult Privacy()
{
    return View();
}
```

Figure 15 - Authorizing logged in Users only

To restrict the access only for admins, the [Authorize(Roles = "admin")] should be used. But in our case, we faced many difficulties to resolve this problem. So we decided to create user *peter_bachour@hotmail.fr* as the only user that can have access to this Page that is the Upload Page to update the database:

```
<div class="collapse navbar-collapse" id="navbarColor01">
  <ul class="navbar-nav nav">
    <li><a asp-area="" asp-controller="Home" asp-action="Index">Home</a>
    <li><a asp-area="" asp-controller="Page" asp-action="Index">Page</a></li>
    <li><a asp-area="" asp-controller="Home" asp-action="About">About us</a></li>
    <li><a asp-area="" asp-controller="Home" asp-action="Contact">Contact us</a></li>

    @if (User.Identity.Name == "peter_bachour@hotmail.fr")
    {
      <li><a asp-area="" asp-controller="Page" asp-action="Upload">Upload</a></li>
    }
  </ul>
  <partial name="_LoginPartial" />
</div>
```

Figure 16 - Restricting this Page only to this User

Hello peter_bachour@hotmail.fr! Logout

Figure 17 - User Logged in

6.3 Database

Since we need to store created items by the user and display them, we need to connect the web application to a database. To avoid writing SQL, the use of Code First will be required.

First, we created the table in SQL Server using Visual Studio through migration.

```
modelBuilder.Entity("OCR.Core.Model.Page", b =>
{
    b.Property<int>("Id")
        .ValueGeneratedOnAdd()
        .HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

    b.Property<string>("FullText");

    b.Property<byte[]>("Image")
        .IsRequired()
        .HasConversion(new ValueConverter<byte[], byte[]>(v => default(byte[]), v => default(byte[]), new ConverterMappingHints(size: 1)))
        .HasColumnType("varbinary(max)");

    b.Property<DateTime>("IssuDate");

    b.Property<int>("IssueNumber");

    b.Property<int>("PageNumber");

    b.HasKey("Id");

    b.ToTable("Page");
});
```

Figure 18 - Migration to SQL Server

Now, since the table is created, it is time to fill it with data using this following model:

```
private async Task<bool> UploadPage(string txtName, string imgName, int pageNumber, CancellationToken ct)
{
    string path = @"C:\Dev\OCR\OCR.WebApi\wwwroot\images\Pages\";
    DateTime issueDate = Convert.ToDateTime("09/27/2018");
    int issueNumber = 42;
    bool result = false;
    if (!_uow.Pages.Get().Any(c => c.PageNumber == pageNumber))
    {
        _uow.Pages.Add(new Page()
        {
            IssuDate = issueDate,
            IssueNumber = issueNumber,
            PageNumber = pageNumber,
            FullText = File.ReadAllText(path + txtName),
            Image = File.ReadAllBytes(path + imgName)
        });
        await _uow.SaveChangesAsync(ct);
        result = true;
    }
    return result;
}

public async Task<bool> UploadPages(List<PageUploadDto> pages, CancellationToken ct)
{
    bool result =
        await UploadPage("4467.txt", "4467.gif", 4467, ct)
        || await UploadPage("4468.txt", "4468.gif", 4468, ct)
        || await UploadPage("4469.txt", "4469.gif", 4469, ct)
        || await UploadPage("4470.txt", "4470.gif", 4470, ct)
        || await UploadPage("4471.txt", "4471.gif", 4471, ct)
        || await UploadPage("4472.txt", "4472.gif", 4472, ct)
        || await UploadPage("4473.txt", "4473.gif", 4473, ct)
        || await UploadPage("4474.txt", "4474.gif", 4474, ct)
        || await UploadPage("4475.txt", "4475.gif", 4475, ct);
    return result;
}
```

Figure 19 - Filling the table

6.4 Swagger

Understanding various methods in a Web API can be challenging for a developer. Here is why we added Swagger that is an OpenAPI. This API solves the problem of generating useful documentation and help pages for Web APIs. It provides benefits such as interactive documentation and API discoverability.

```
app.UseSwagger();
app.UseSwaggerUI(c =>
{
    c.RoutePrefix = "help";
    c.SwaggerEndpoint("../swagger/v1/swagger.json", "My API V1");
    c.InjectStylesheet("../css/swagger.min.css");
});
```

Figure 20 - Swagger in ConfigureServices

6.1 Facebook and Twitter API

Adding Sharing APIs was one of our targets first in the project because it can help the user to share with other people on Social Media platforms the Pages he found interesting. So we used Facebook API and Twitter API and it can only be accessed to signed in Users.

```
@if (SignInManager.IsSignedIn(User)) {
<div style="float:right">
    <ul>
        <a class="twitter-share-button" target="_blank" data-layout="button"
            href="https://twitter.com/intent/tweet?text=Just%20Found%20the%20best%20page%20for%20the%20Official%20Lebanese%20Journal"
            data-size="large">Tweet</a>
    </ul>
    <ul>
        <a class="fb-share-button" data-href="https://www.google.com" data-layout="button" data-size="large">Share</a>
    </ul>
</div>
}
<script async src="https://platform.twitter.com/widgets.js" charset="utf-8"></script>
<script>
    (function (d, s, id) {
        var js, fjs = d.getElementsByTagName(s)[0];
        if (d.getElementById(id)) return;
        js = d.createElement(s); js.id = id;
        js.src = 'https://connect.facebook.net/en_US/sdk.js#xfbml=1&version=v3.2';
        fjs.parentNode.insertBefore(js, fjs);
    })(document, 'script');
```

Figure 21 - Facebook and Twitter APIs

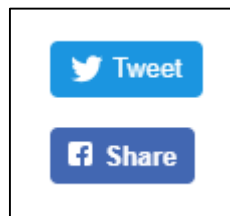


Figure 22 - Facebook and Twitter Buttons

7 Discussion and Future Work

7.1 Discussion

7.1.1 Difficulty 1

Integrating a full OCR API was our main objective until we realized that the problem was with the NuGet Packages. The package was either paid or was not compatible with .Net Core or could not support Arabic.

7.1.2 Difficulty 2

Inserting data in the database was a big challenge since we cannot insert images into the tables. So, we found the solution of converting the images to varBinary type. We tried to use the paths of the pictures but since we are connecting to a server, we could lose data.

7.1.3 Difficulty 3

Finding a solution to Authorize only Users identified as Admin to access some pages was a big challenge that we could not resolve. But in order to continue the project we found a way to get around the problem.

7.2 Future Work

Here are some additional features that could be added to the project to improve its usability and move it closer to a product that can be good in the market.

7.2.1 User Interface

The user interface could be updated to be more ecstatically satisfying as well as more modern. The design of the Website is somewhat classic and serious.

7.2.2 Favorite Pages

In order to help the user to check a page more than once, we can create a section to make him able to save pages for a further read.

8 Conclusion

The main goal of the project was to put in place a web application where we can search in the Official Lebanese Journal. After that, this web application was to be linked to the official website of the Lebanese Journal where the Pages are published weekly and then pass these pages to an OCR system and finally add them to our database where users can easily search in them.

9 Glossary

- API: application programming interface.
- Mockup: A model of a machine or structure used for experimental purposes.
- HTML: Hypertext markup language is the standard markup language for creating Webpages and WebApps.
- CSS: cascading style sheets that is used for describing the presentation of a document written in a markup language like HTML.
- CSHTML: C# based razor view engine in addition to straight HTML.